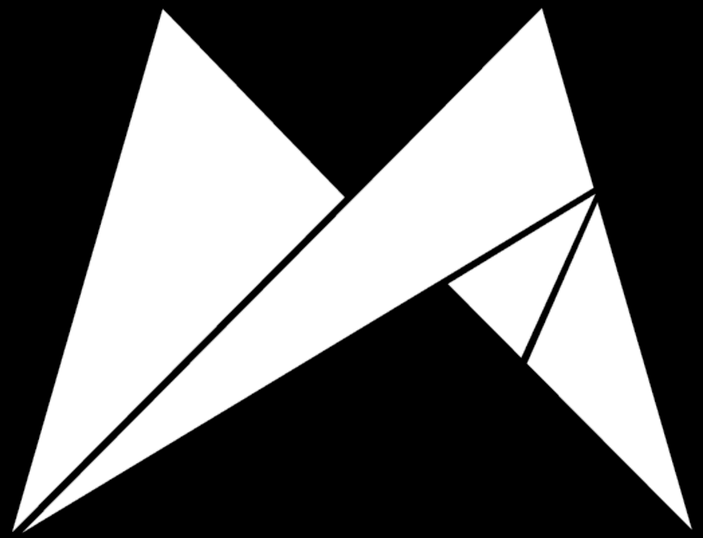
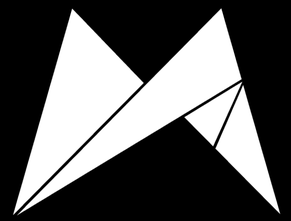


Mobile Academy



Mobile
Academy

@eldudi

#TDDCraftConf

What is a unit test?

“Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use.”

Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices in Software Management.

Um… what?

“Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use.”

Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices in Software Management.

What is an app?

An app is a set of behaviours created by programmer and expected by user.

We, programmers, have a limited cognition. As all humans do.

We can’t always ‘load’
all of the code of our
app into our memory.

This means that we can, by accident, change the behaviour of the app.

Preserving behaviour of complex systems is hard. In fact, of any system at all.

Enter unit tests.

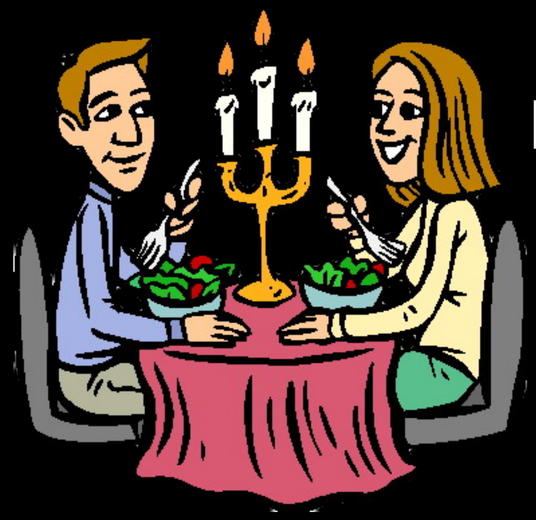
Unit test is a failsafe to make sure app behaviour is preserved.

What is a unit test?

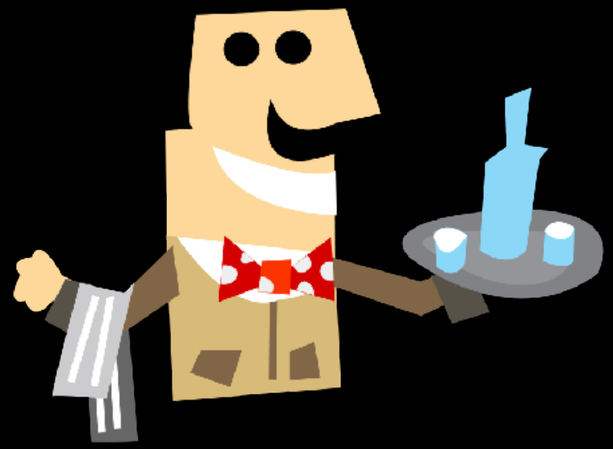
Unit tests test smallest parts of your code in isolation with test code

Unit tests test smallest parts of your code in **isolation** with test code

Test isolation



processOrder

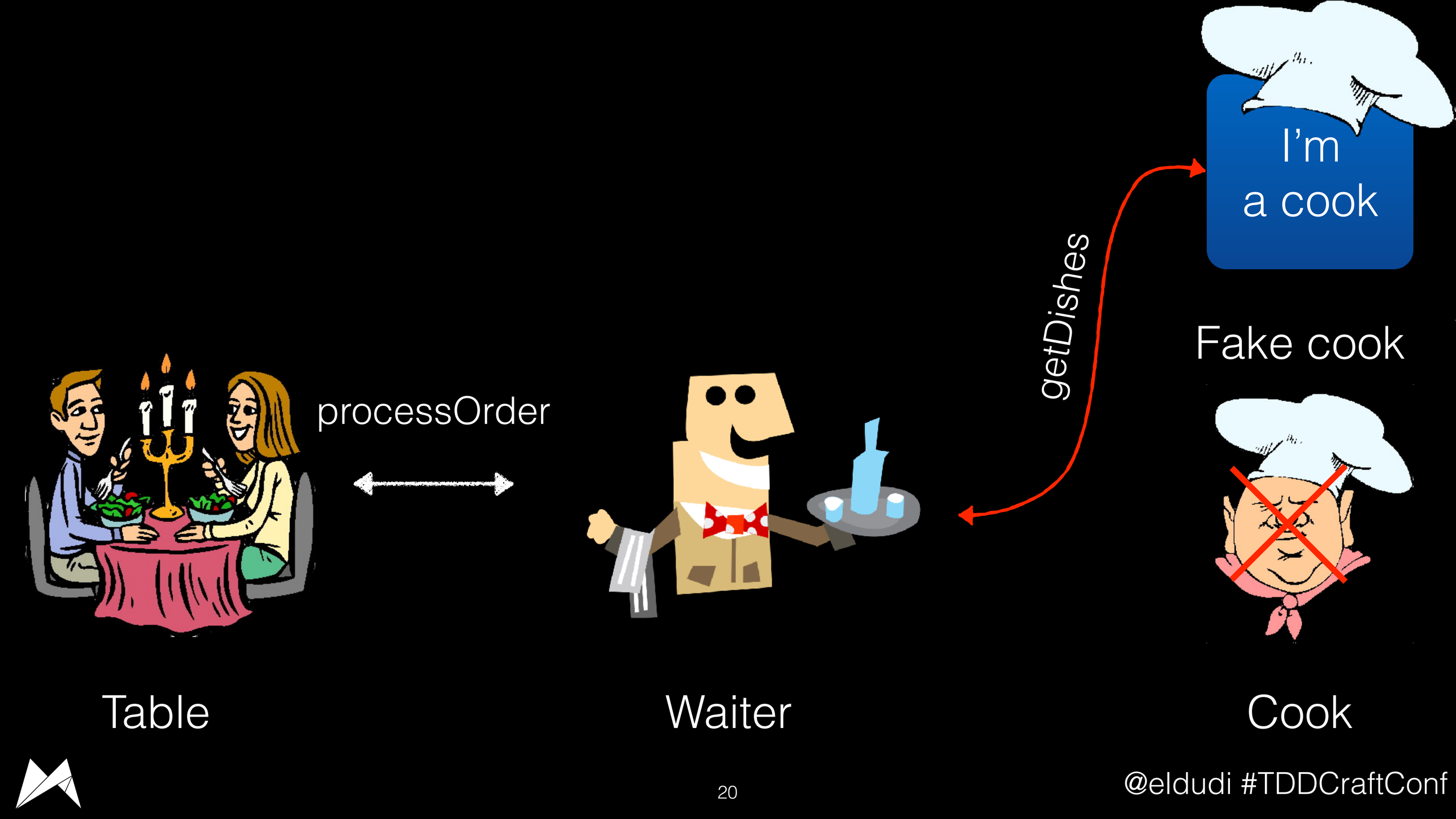


getDishes

Table

Waiter

Cook


I'm
a cook
Fake cook
getDishes
processOrder
Table
Waiter
Cook

Why isolate?

Unit test lifecycle

Unit test lifecycle

- Arrange
- Act
- Assert

When a unit test is not a unit test?

A test is not a unit test if…

- It talks to a database
- It communicates across network
- It touches the file system
- You have to do special things to your environment to run it (edit config files etc)

A test is not a unit test if…

- It talks to a database
- It communicates across network
- It touches the file system
- You have to do special things to your environment to run it (edit config files etc)

A test is not a unit test if…

- It talks to a database
- It communicates across network
- It touches the file system
- You have to do special things to your environment to run it (edit config files etc)

A 100 ms tests is a
very slow test.

1500 tests each running 100 ms. That’s 150 seconds. Two and a half minutes.

Where does TDD fit in all this?

Test Driven Development

Test **Driven** Development

In TDD you always
write test first. Always.

TDD is not “just adding tests first”. It’s a complete workflow.

TDD is a great way to determine how complex your code has become.

You just have to listen.

Have to fake seven objects to isolate test?

Have to inject a fake into a fake into a fake?

Your test setup
method has 70 lines?

You need to simulate five events to test one method?

This always points to an overcomplicated design.

And your tests are here to point that out. Very clearly.

By writing the test first, you're forced into thinking what responsibilities given object should have.

By writing test first you're becoming a consumer of your upcoming API.

Clarify requirements

All behaviors are testable.
The only thing that is not testable is your code

What unit tests can’t do?

Unit tests are never a guarantee that you won’t ship a bug.

But they’re damn good at greatly reducing amount of bugs. And time spent on QA.

Are unit tests an invaluable tool for writing great software? Heck yes. Am I going to produce a poor product if I can't unit test? Hell no.

Jonathan Rasmusson

http://agilewarrior.wordpress.com/2012/10/06/its-not-about-the-unit-tests/

Quick

Quick

BDD Testing Framework

Behavior Driven Development
Test Driven Development

BDD aims to improve certain aspect of TDD

BDD tries to help you
know **what** to test

When writing tests
don’t think ‘tests’

Think about ‘behaviors’

Think about examples
how your object should
behave

Examples should cover only the interface of your object

Only the interface

Good habits

- Work outside-in
- Use examples to clarify requirements
- Use ubiquitous language

Technical stuff now

Quick

Based on XCTest

Minimalistic implementation

Syntax

```
class DolphinSpec: QuickSpec {
    override func spec() {
        it(“should be friendly") {
            expect(Dolphin().isFriendly).to(beTruthy())
        }

        it(“should be smart") {
            expect(Dolphin().isSmart).to(beTruthy())
        }
    }
}
```

Describe/Context closures

Used to make tests more readable.

And isolate behaviour for different scenarios.

```
class DolphinSpec: QuickSpec {
    override func spec() {
        describe("dolphin") {
            describe("its click") {
                it("should be loud") {
                    let click = Dolphin().click()
                    expect(click.isLoud).to(beTruthy())
                }

                it("should have a high frequency") {
                    let click = Dolphin().click()
                    expect(click.hasHighFrequency).to(beTruthy())
                }
            }
        }
    }
}
```

You can have as many nested describes as you want.

Before/After each closures

```
var appDelegate: AppDelegate!

beforeEach {
    appDelegate = AppDelegate()
}

afterEach {
   appDelegate = nil
}

it("should have a window”) {
   expect(appDelegate.window).to(beKindOf(UIWindow))
}
```

```
var appDelegate: AppDelegate!

beforeEach {
    appDelegate = AppDelegate()
}


it("should have a window”) {
    expect(appDelegate.window).to(beKindOf(UIWindow))
}

afterEach {
    appDelegate = nil
}
```

1

2

3

Let’s write our very first unit test!

Hands on!

Configuring tests

Focusing tests

Focusing tests

```
fdescribe("Example specs on NSString") {

fit("lowercaseString returns a new string with
everything in lower case") {

fcontext("init with damping") {
```

PENDING

PENDING

```
pending("lowercaseString returns a new string with
everything in lower case") {}
```

x’ing tests

x’ing tests

```
xdescribe("Example specs on NSString") {

xit("lowercaseString returns a new string with
everything in lower case") {

xcontext("init with damping") {
```

Unit tests results

Unit tests results

How to understand the output?

Xcode, AppCode, Command Line

All give the same results. Devil is in the details

```
(…)

-[SpecSuiteName passing_spec_name]
  Test Case '-[SpecSuiteName passing_spec_name]' started.
  Test Case '-[SpecSuiteName passing_spec_name]' passed
(0.271 seconds).

-[SpecSuiteName failling_spec_name]
  Test Case '-[SpecSuiteName failling_spec_name]' started.
  Test Case '-[SpecSuiteName failling_spec_name]' failed
(0.002 seconds).

(…)

Executed 2 tests, with 1 failure (1 unexpected) in 0.273
(0.278) seconds

2 tests; 0 skipped; 1 failure; 1 exception; 0 pending
```

```
(…)

-[SpecSuiteName passing_spec_name]
  Test Case '-[SpecSuiteName passing_spec_name]' started.
  Test Case '-[SpecSuiteName passing_spec_name]' passed
(0.271 seconds).

-[SpecSuiteName failling_spec_name]
  Test Case '-[SpecSuiteName failling_spec_name]' started.
  Test Case '-[SpecSuiteName failling_spec_name]' failed
(0.002 seconds).

(…)

Executed 2 tests, with 1 failure (1 unexpected) in 0.273
(0.278) seconds

2 tests; 0 skipped; 1 failure; 1 exception; 0 pending
```

```
(…)

-[SpecSuiteName passing_spec_name]
  Test Case '-[SpecSuiteName passing_spec_name]' started.
  Test Case '-[SpecSuiteName passing_spec_name]' passed
(0.271 seconds).

-[SpecSuiteName failling_spec_name]
  Test Case '-[SpecSuiteName failling_spec_name]' started.
  Test Case '-[SpecSuiteName failling_spec_name]' failed
(0.002 seconds).

(…)

Executed 2 tests, with 1 failure (1 unexpected) in 0.273
(0.278) seconds

2 tests; 0 skipped; 1 failure; 1 exception; 0 pending
```

```
(…)

-[SpecSuiteName passing_spec_name]
  Test Case '-[SpecSuiteName passing_spec_name]' started.
  Test Case '-[SpecSuiteName passing_spec_name]' passed
(0.271 seconds).

-[SpecSuiteName failling_spec_name]
  Test Case '-[SpecSuiteName failling_spec_name]' started.
  Test Case '-[SpecSuiteName failling_spec_name]' failed
(0.002 seconds).

(…)

Executed 2 tests, with 1 failure (1 unexpected) in 0.273
(0.278) seconds

2 tests; 0 skipped; 1 failure; 1 exception; 0 pending
```

```
(…)

-[SpecSuiteName passing_spec_name]
  Test Case '-[SpecSuiteName passing_spec_name]' started.
  Test Case '-[SpecSuiteName passing_spec_name]' passed
(0.271 seconds).

-[SpecSuiteName failling_spec_name]
  Test Case '-[SpecSuiteName failling_spec_name]' started.
  Test Case '-[SpecSuiteName failling_spec_name]' failed
(0.002 seconds).

(…)

Executed 2 tests, with 1 failure (1 unexpected) in 0.273
(0.278) seconds

2 tests; 0 skipped; 1 failure; 1 exception; 0 pending
```

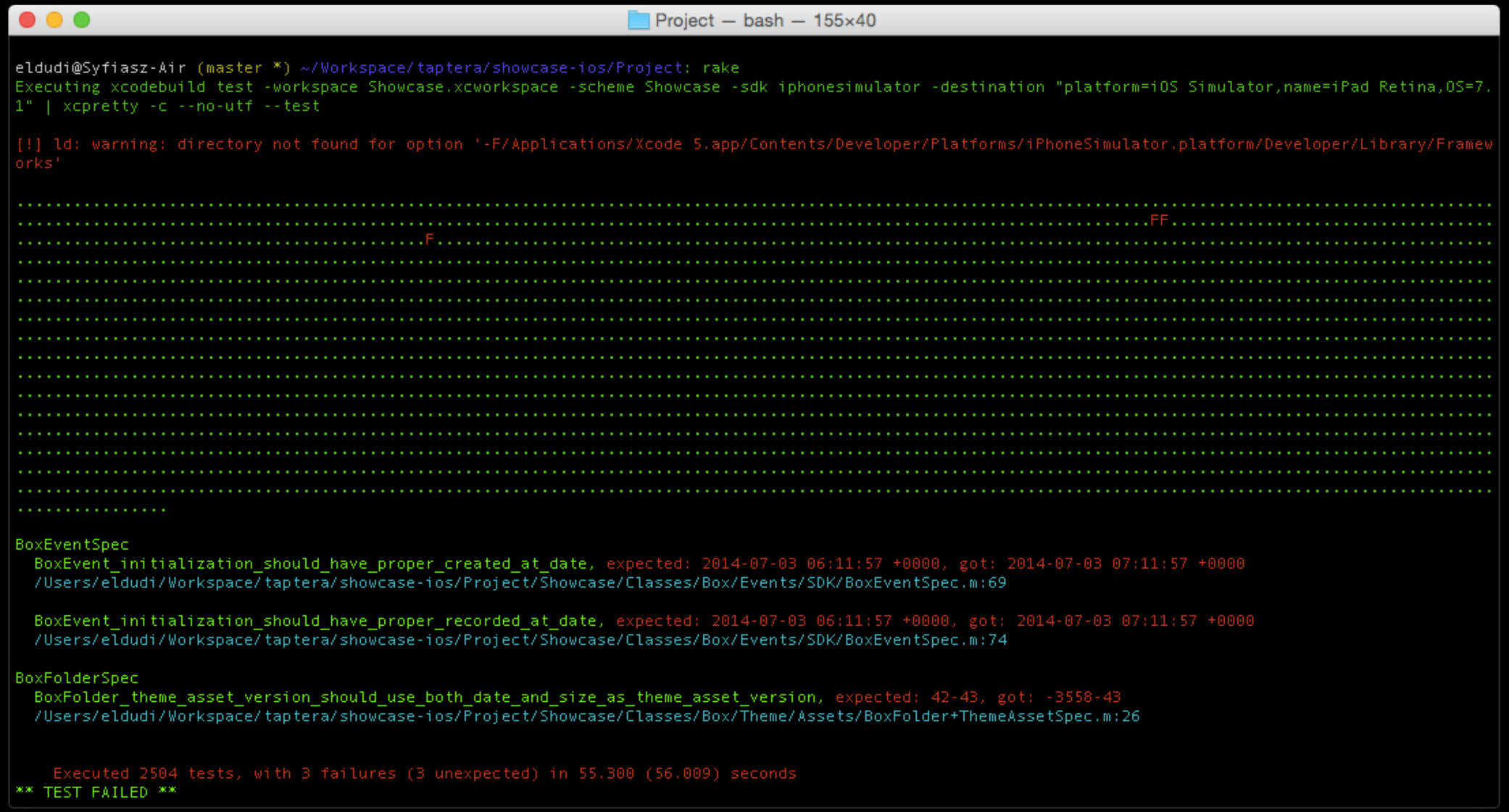
Run your tests from command line.

Seriously, do.
It’s pretty awesome.

“Perfect” setup:
Have your tests run each time you change something in a file.

Enhance your tests output.

```
(…)

-[SpecSuiteName passing_spec_name]
  Test Case '-[SpecSuiteName passing_spec_name]' started.
  Test Case '-[SpecSuiteName passing_spec_name]' passed
(0.271 seconds).

-[SpecSuiteName failling_spec_name]
  Test Case '-[SpecSuiteName failling_spec_name]' started.
  Test Case '-[SpecSuiteName failling_spec_name]' failed
(0.002 seconds).

(…)

Executed 2 tests, with 1 failure (1 unexpected) in 0.273
(0.278) seconds

2 tests; 0 skipped; 1 failure; 1 exception; 0 pending
```


Project — bash — 155×40
eldudi@Syfiasz-Air (master *) ~/Workspace/taptera/showcase-ios/Project: rake
Executing xcodebuild test -workspace Showcase.xcworkspace -scheme Showcase -sdk iphonesimulator -destination "platform=iOS Simulator,name=iPad Retina,OS=7.1" | xcpretty -c --no-utf --test
[!] ld: warning: directory not found for option '-F/Applications/Xcode 5.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/Library/Frameworks'
BoxEventSpec
BoxEvent_initialization_should_have_proper_created_at_date, expected: 2014-07-03 06:11:57 +0000, got: 2014-07-03 07:11:57 +0000
/Users/eldudi/Workspace/taptera/showcase-ios/Project/Showcase/Classes/Box/Events/SDK/BoxEventSpec.m:69
BoxEvent_initialization_should_have_proper_recorded_at_date, expected: 2014-07-03 06:11:57 +0000, got: 2014-07-03 07:11:57 +0000
/Users/eldudi/Workspace/taptera/showcase-ios/Project/Showcase/Classes/Box/Events/SDK/BoxEventSpec.m:74
BoxFolderSpec
BoxFolder_theme_asset_version_should_use_both_date_and_size_as_theme_asset_version, expected: 42-43, got: -3558-43
/Users/eldudi/Workspace/taptera/showcase-ios/Project/Showcase/Classes/Box/Theme/Assets/BoxFolder+ThemeAssetSpec.m:26
Executed 2504 tests, with 3 failures (3 unexpected) in 55.300 (56.009) seconds
** TEST FAILED **

Resources & Contact

Code Examples

github.com/mobile-academy/ios-tdd-swift-craft-conf

Contact

@eldudi

pawel@dudek.mobi